Huge integer division

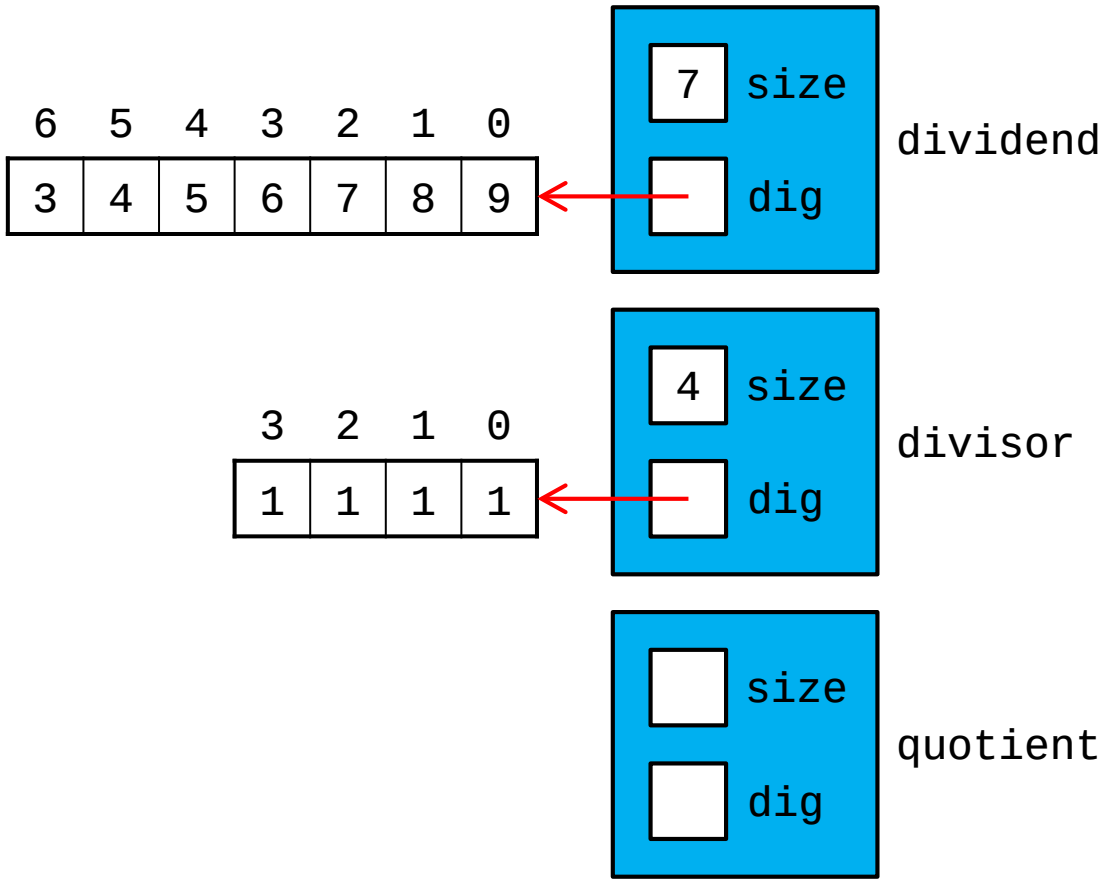
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```

```
int main()
{
   char strA[ 251 ], strB[ 251 ];

   int T;
   cin >> T;
   for( int t = 0; t < T; ++t )
   {
      cin >> strA >> strB;

      HugeInt dividend;
      dividend.size = strlen( strA );
      dividend.digit = new int[ dividend.size ]();
      for( int i = 0; i < dividend.size; ++i )
         dividend.digit[ i ] = strA[ dividend.size - 1 - i ] - '0';

      HugeInt divisor;
      divisor.size = strlen( strB );
      divisor.digit = new int[ divisor.size ]();
      for( int i = 0; i < divisor.size; ++i )
         divisor.digit[ i ] = strB[ divisor.size - 1 - i ] - '0';
```

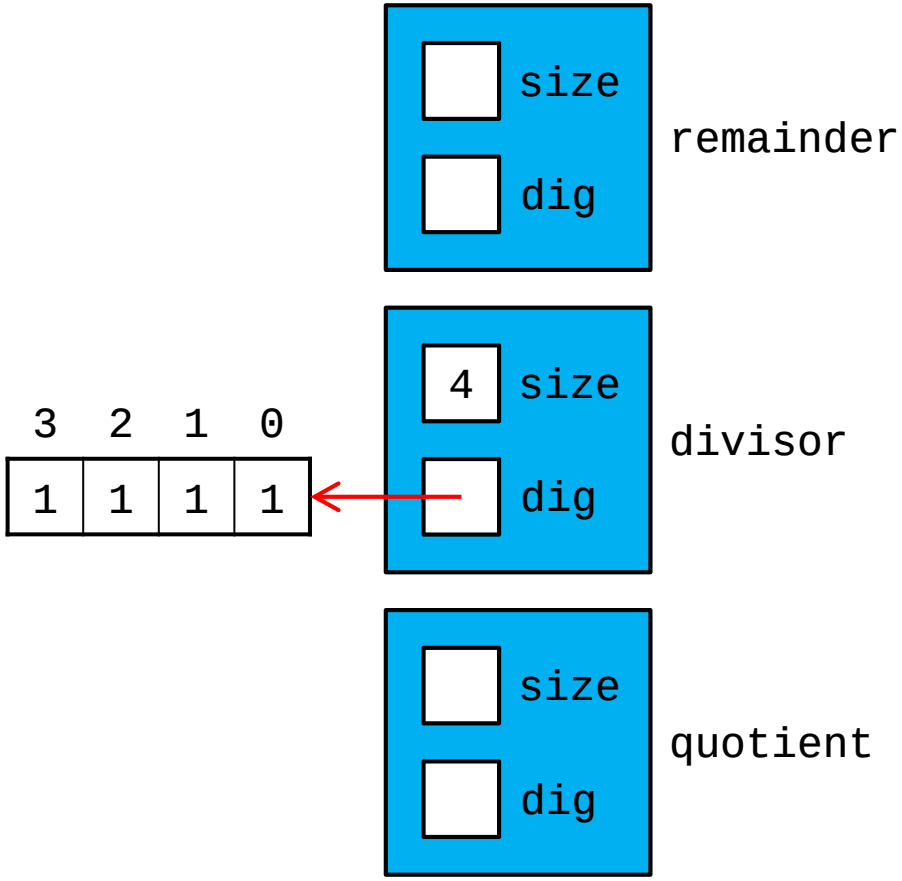
```
        HugeInt quotient;
        HugeInt remainder;
        division( dividend, divisor, quotient, remainder );

        for( int i = quotient.size - 1; i >= 0; i-- )
           cout << quotient.digit[ i ];
        cout << endl;

        for( int i = remainder.size - 1; i >= 0; i-- )
           cout << remainder.digit[ i ];
        cout << endl;

        delete[] dividend.digit;
        delete[] divisor.digit;
        delete[] quotient.digit;
        delete[] remainder.digit;
    }
}
```

```
struct HugeInt
{
   int size;
   int *digit;
};

cin >> strA >> strB;

HugeInt dividend;
dividend.size = strlen( strA );
dividend.digit = new int[ dividend.size ]();
for( int i = 0; i < dividend.size; ++i )
   dividend.digit[ i ] = strA[ dividend.size - 1 - i ] - '0';

HugeInt divisor;
divisor.size = strlen( strB );
divisor.digit = new int[ divisor.size ]();
for( int i = 0; i < divisor.size; ++i )
   divisor.digit[ i ] = strB[ divisor.size - 1 - i ] - '0';

HugeInt quotient;
HugeInt remainder;
division( dividend, divisor, quotient, remainder );
```

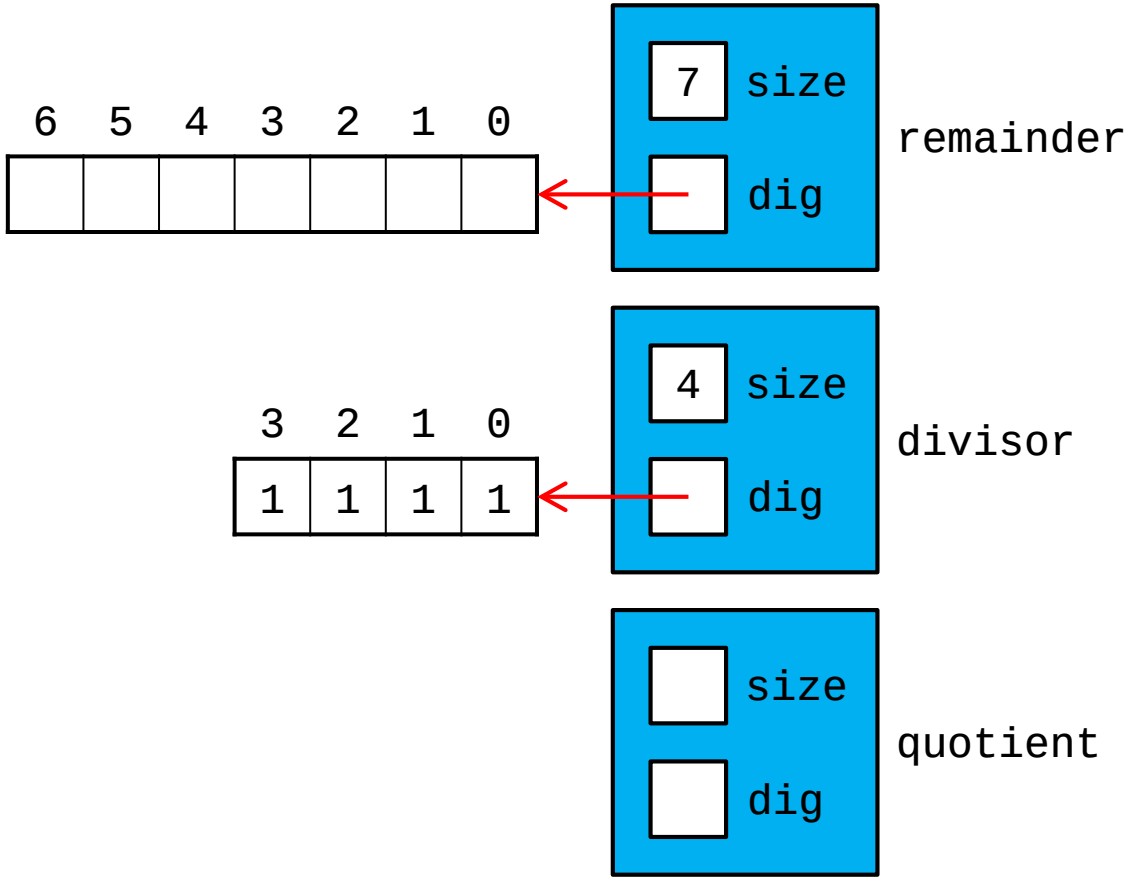
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



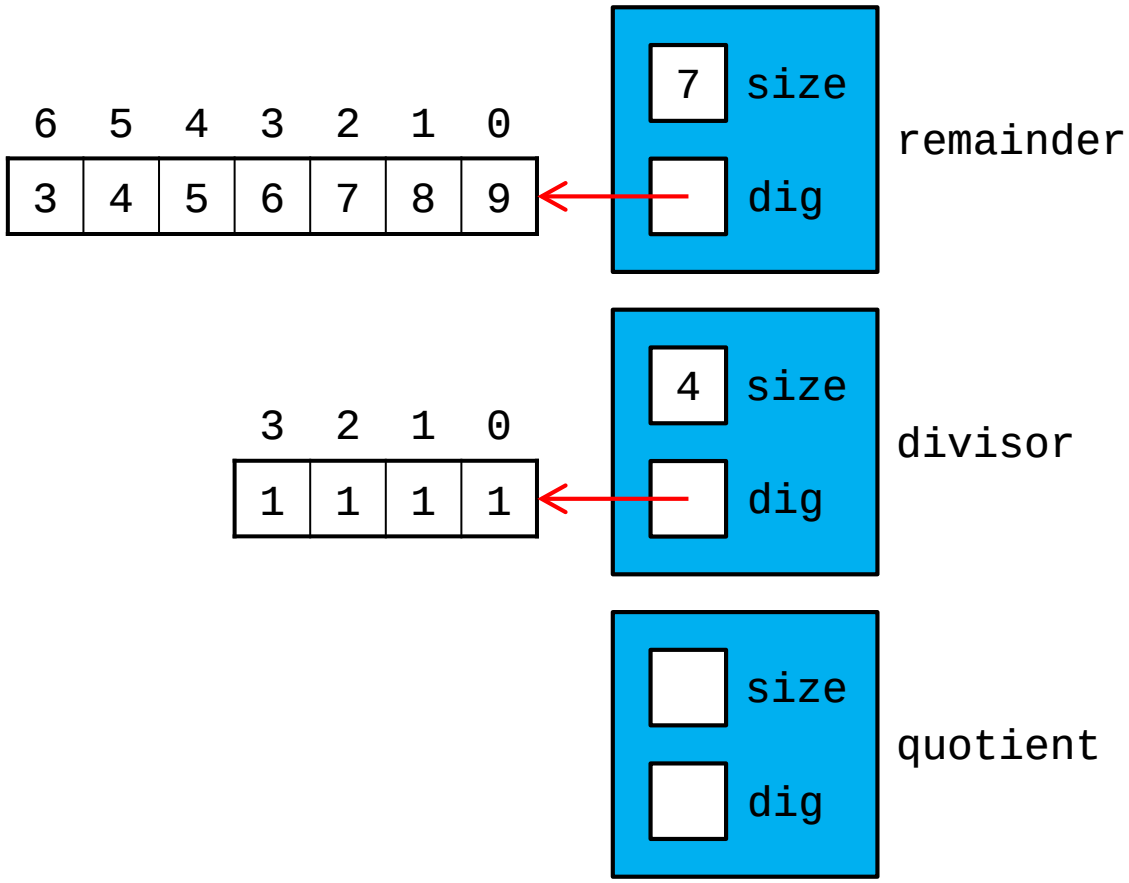
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



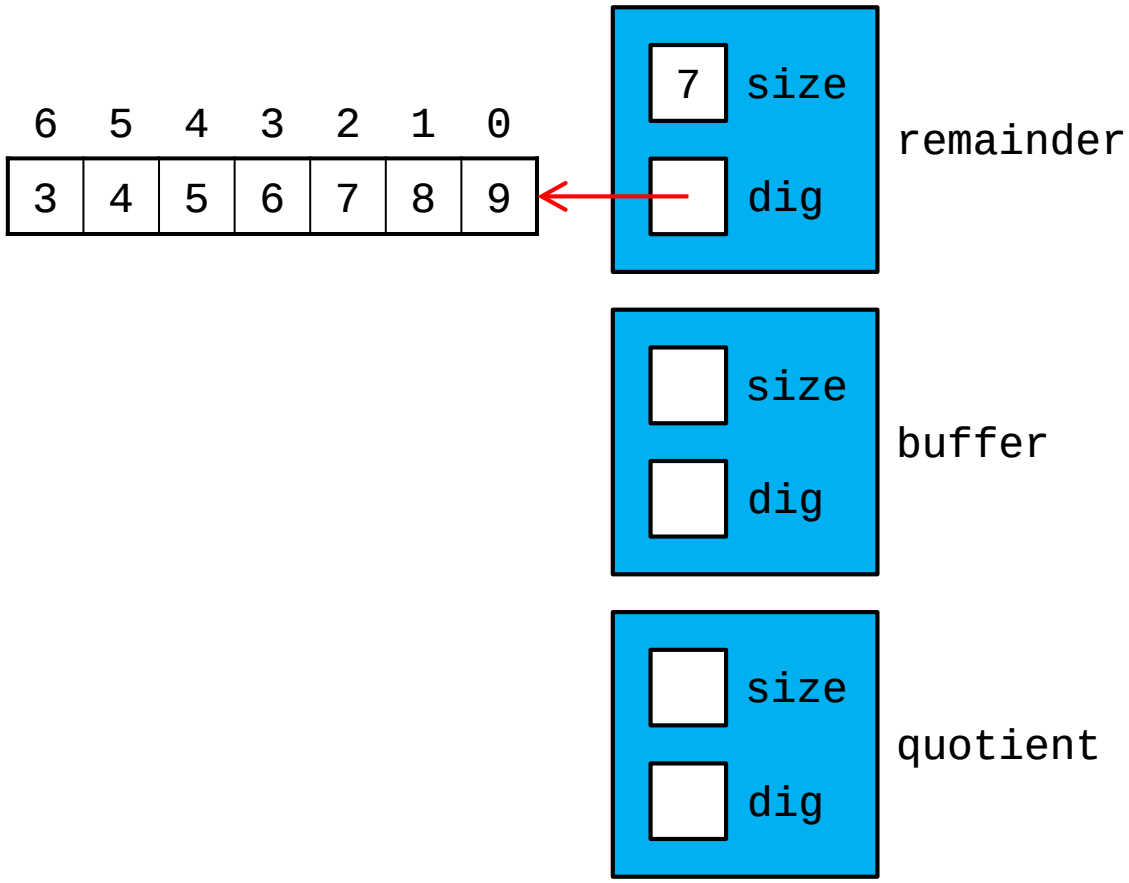
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



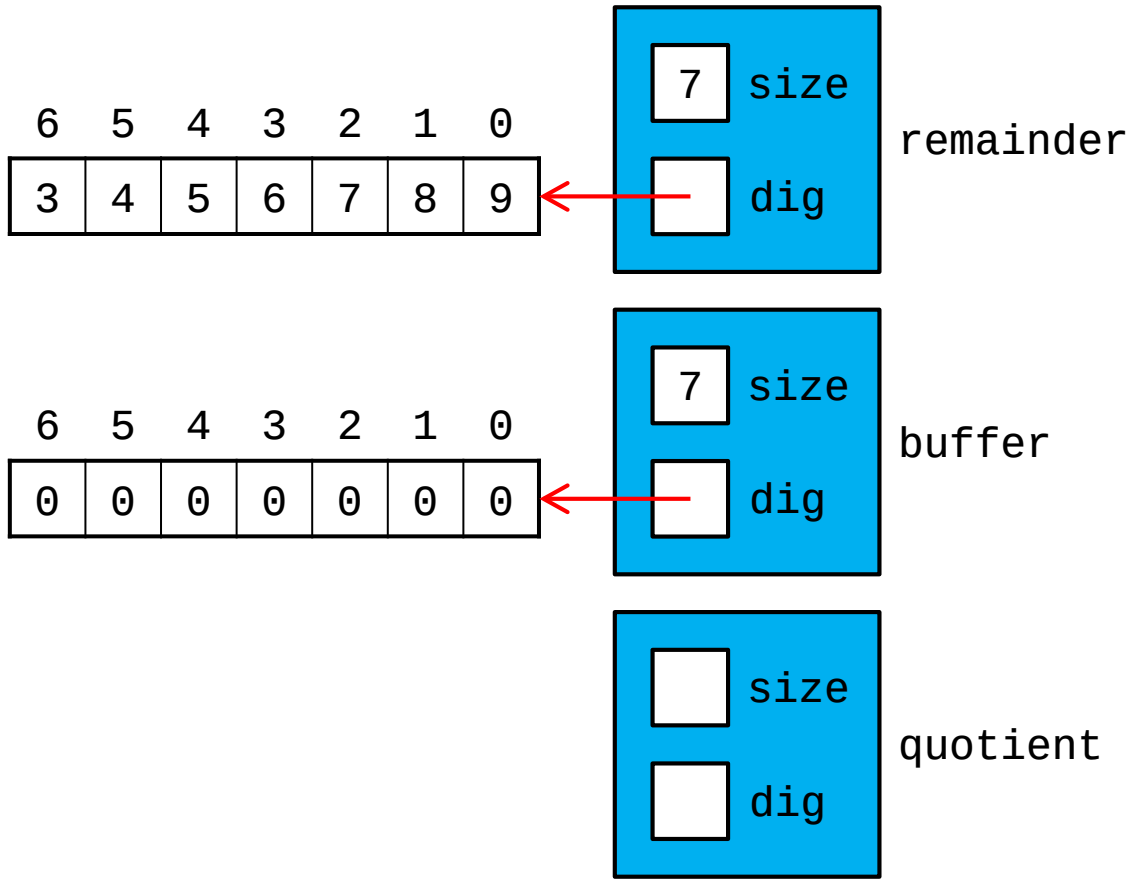
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



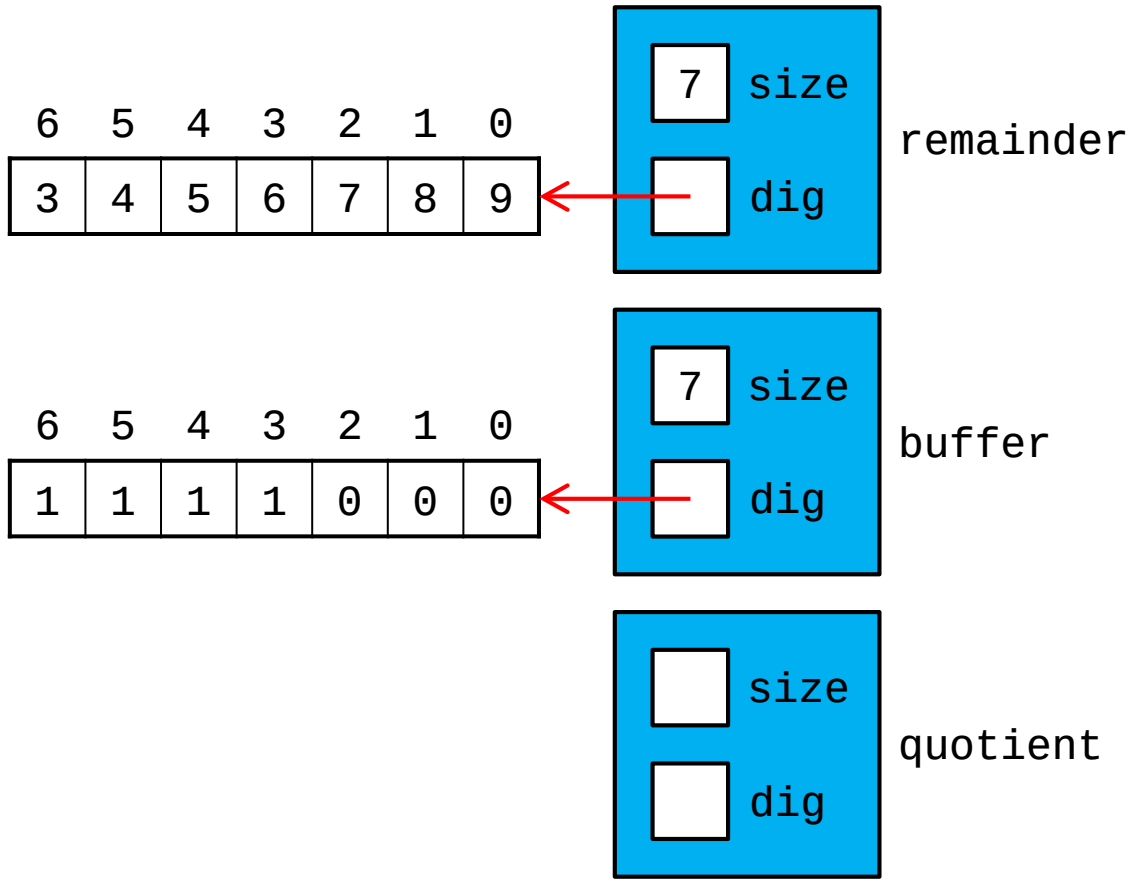
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



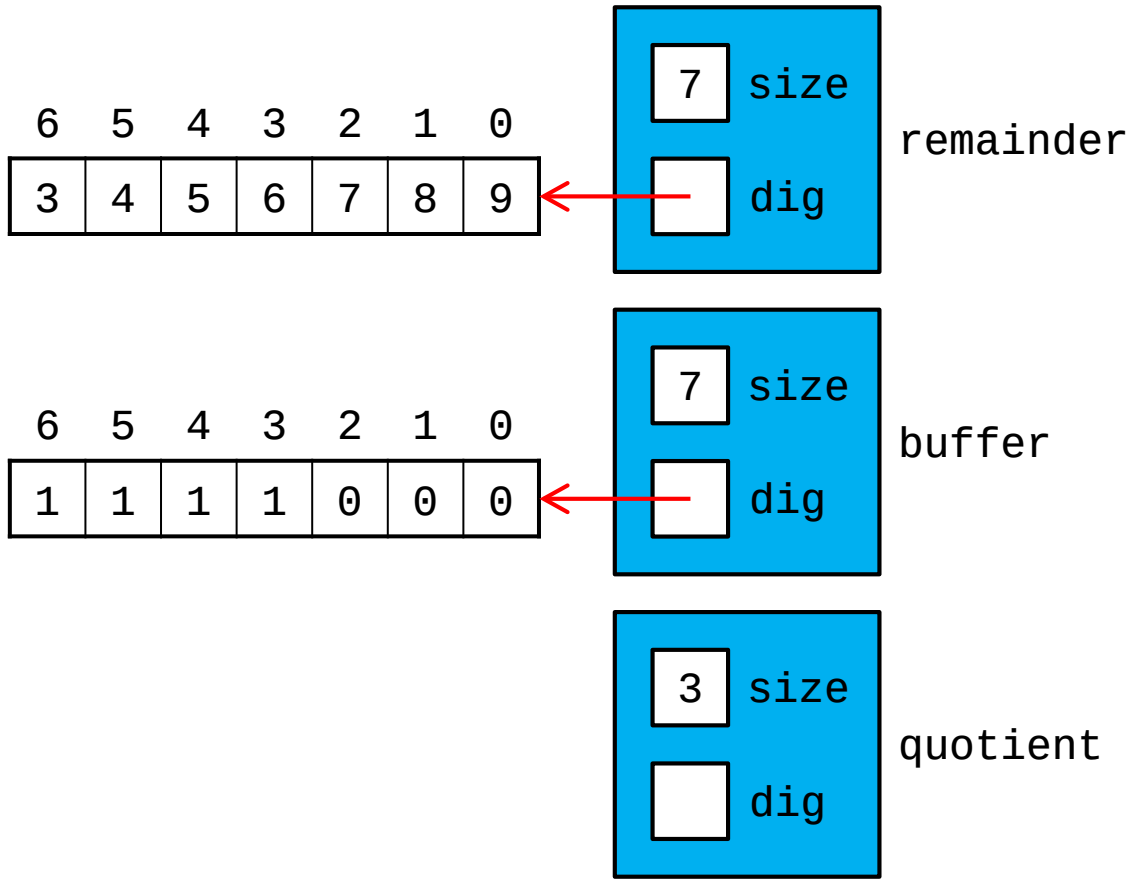
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



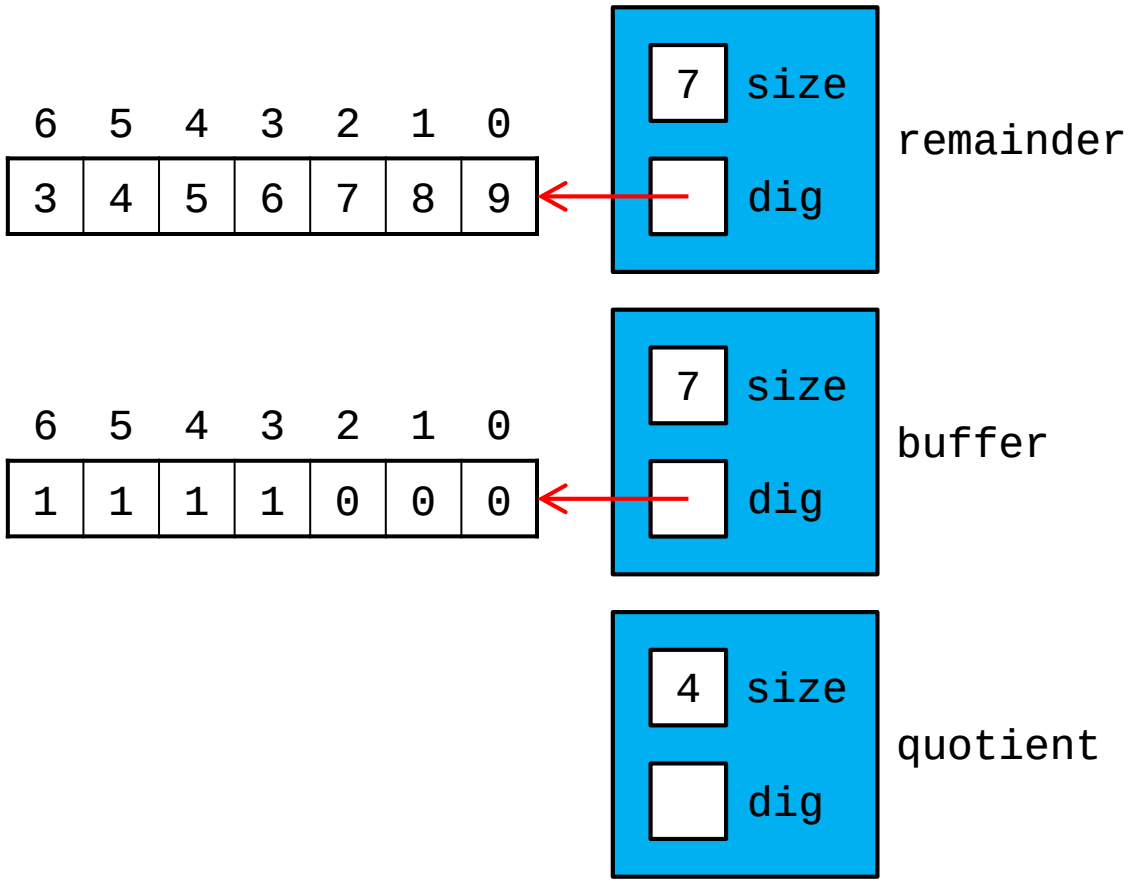
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



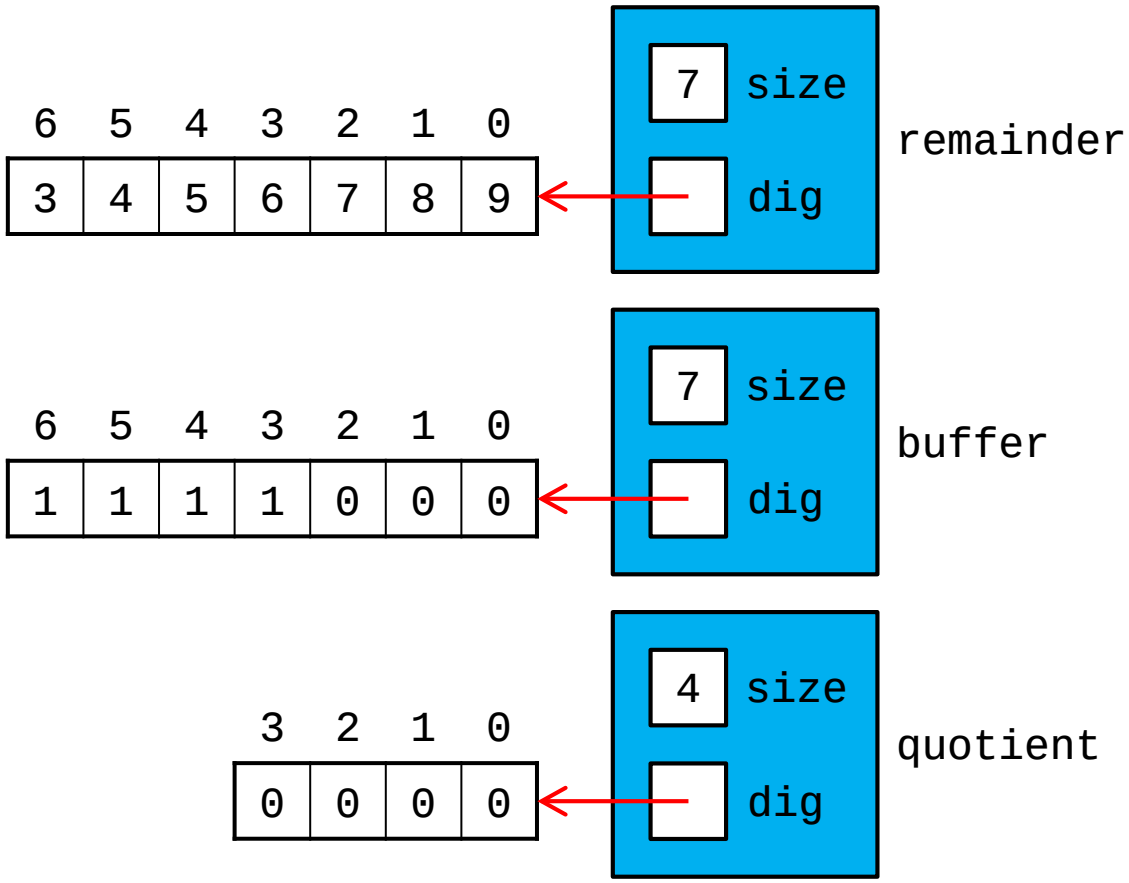
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



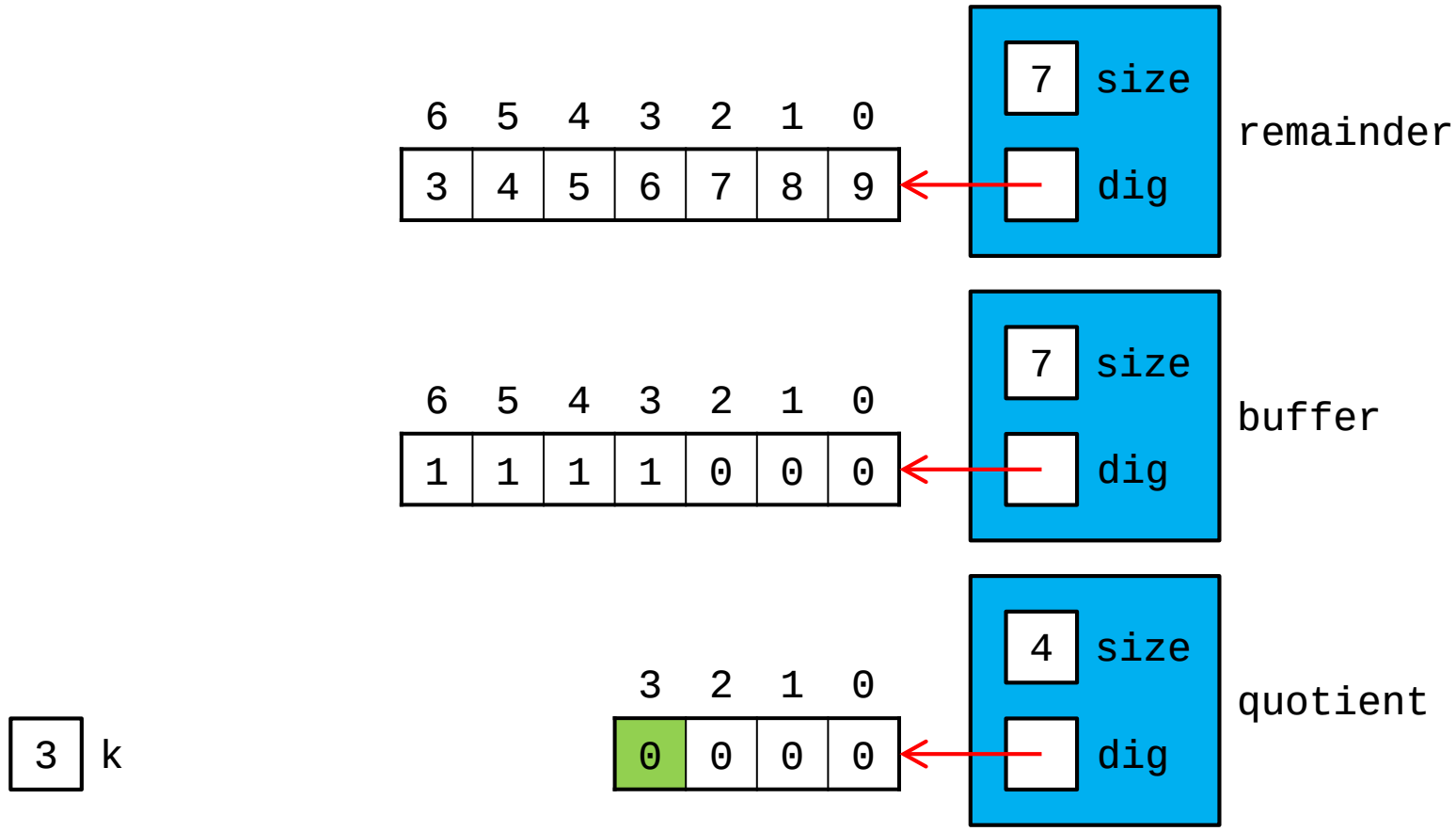
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



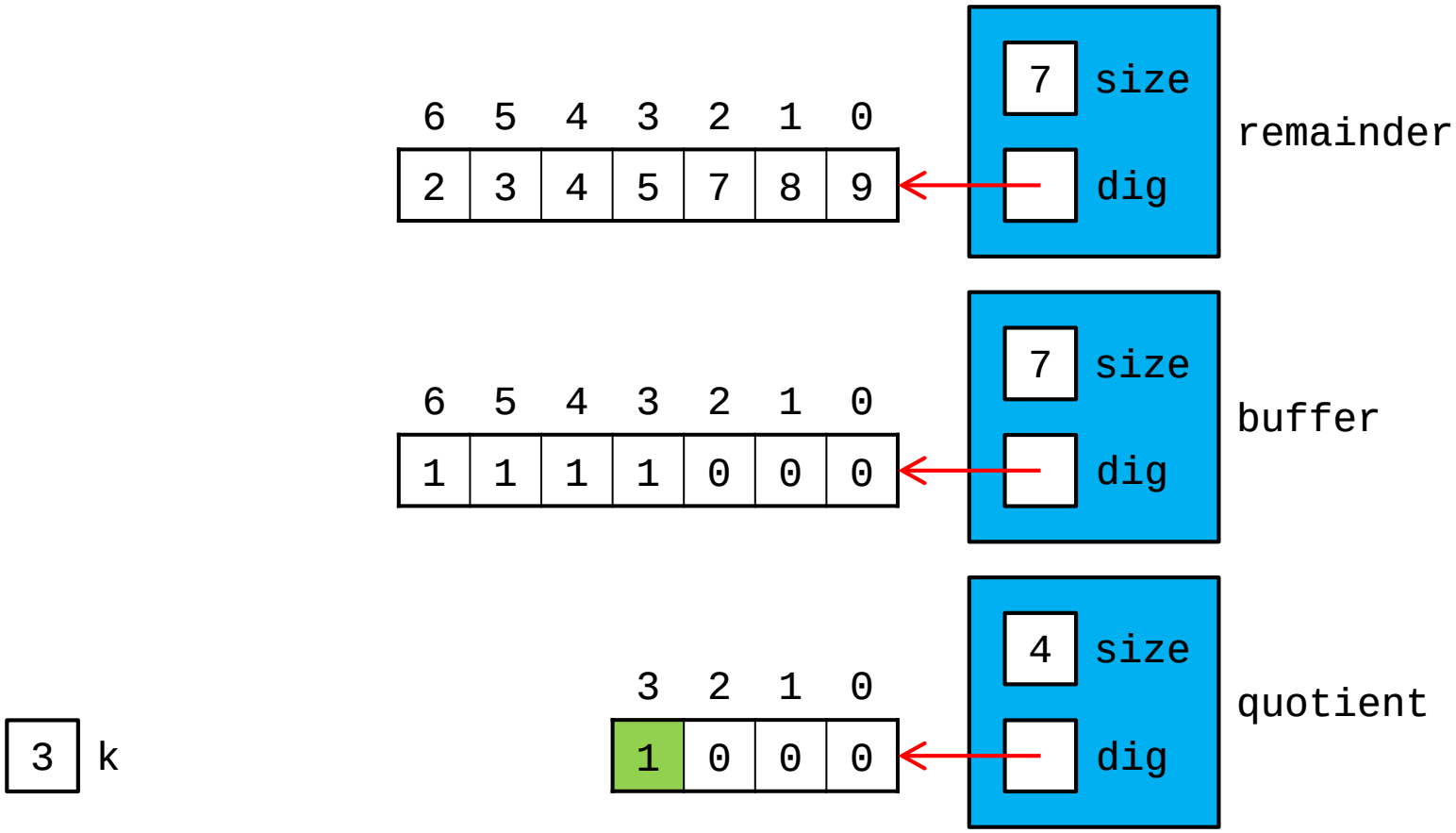
```
remainder = dividend
n = dividend.size – divisor.size
buffer = divisor shift left by n positions
quotient.size = n
if( dividend < buffer )
   buffer = buffer / 10 // divideBy10( buffer )
else
   quotient.size++
quotient = 0;
```



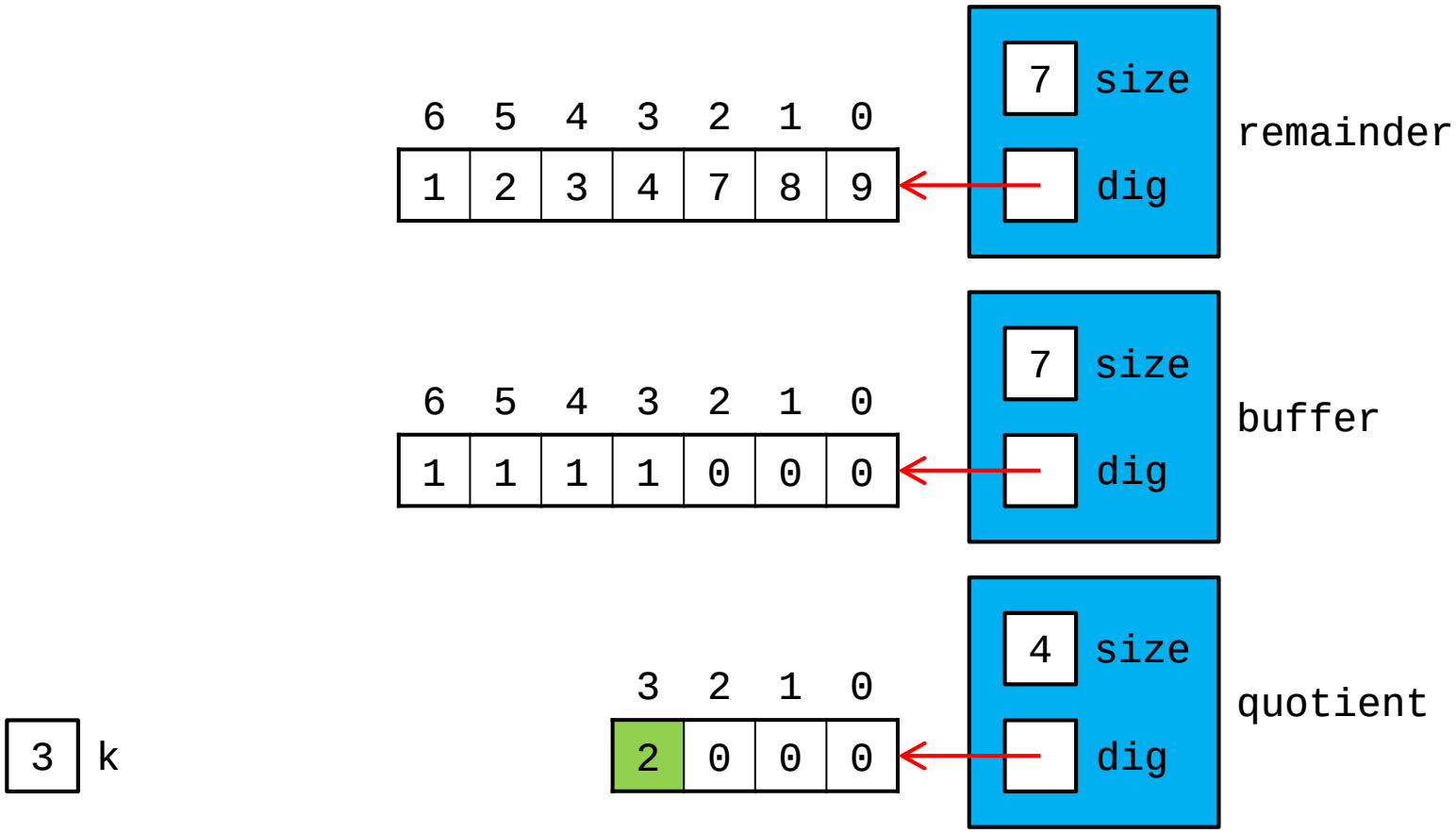
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



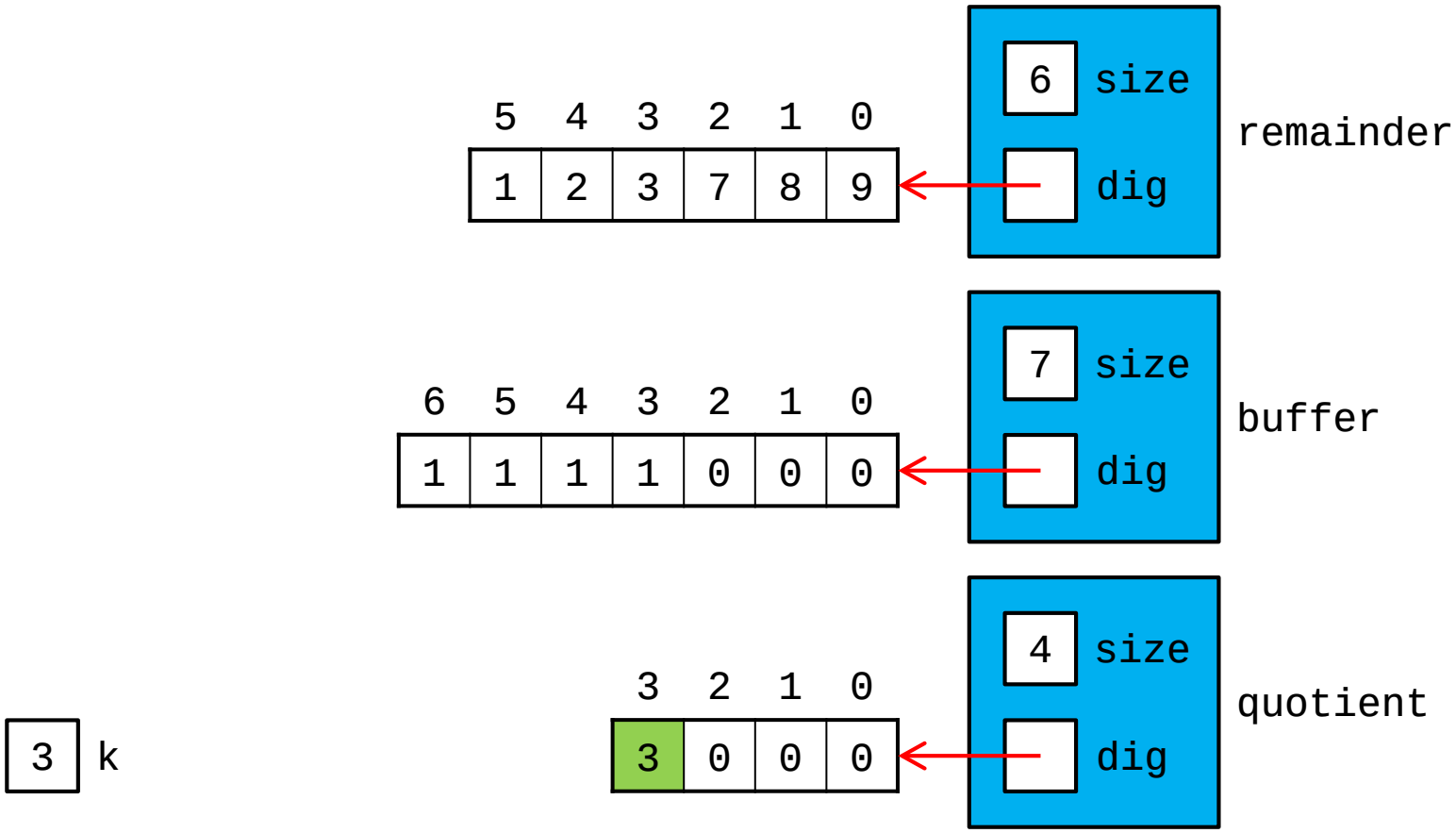
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



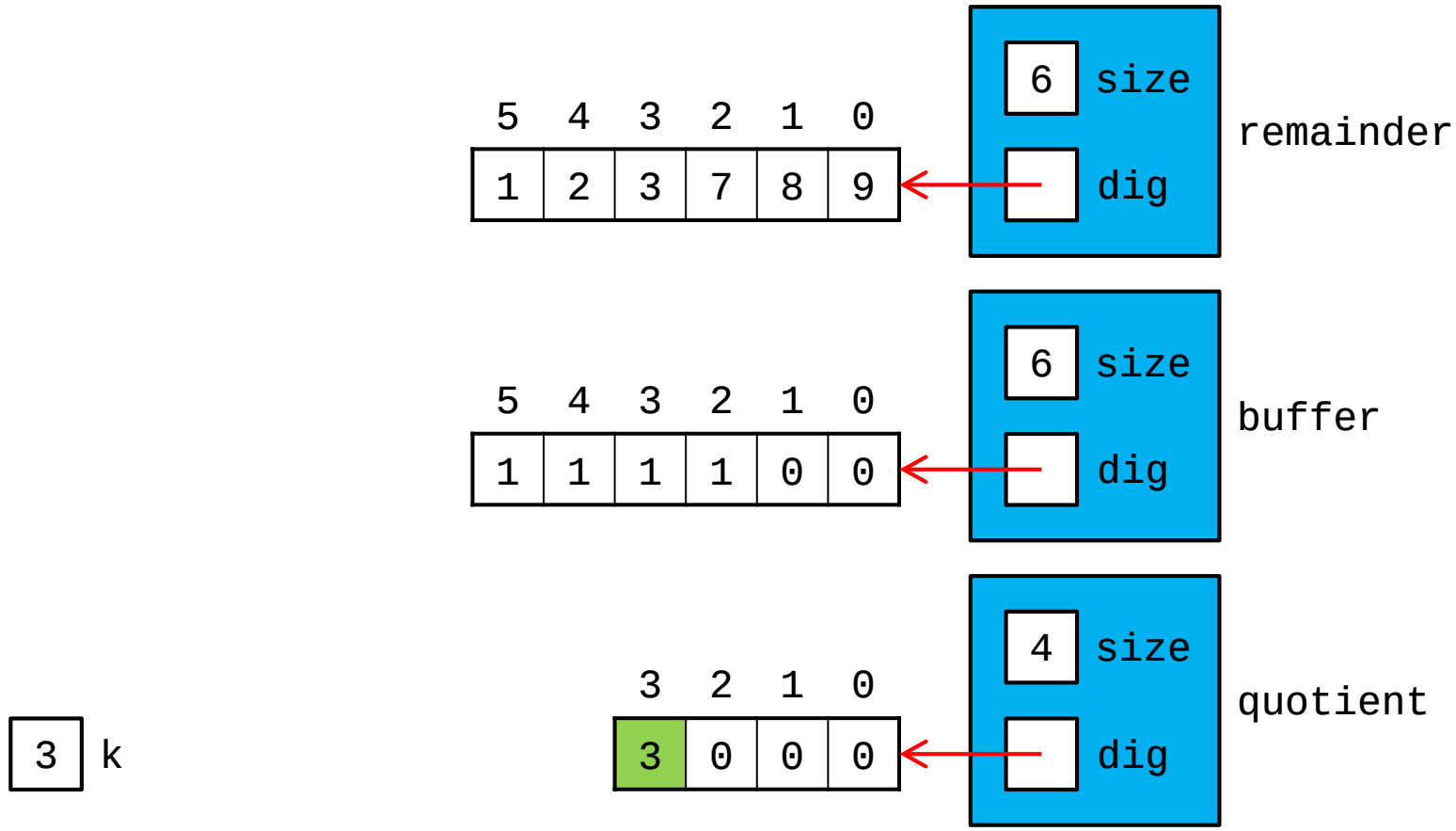
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



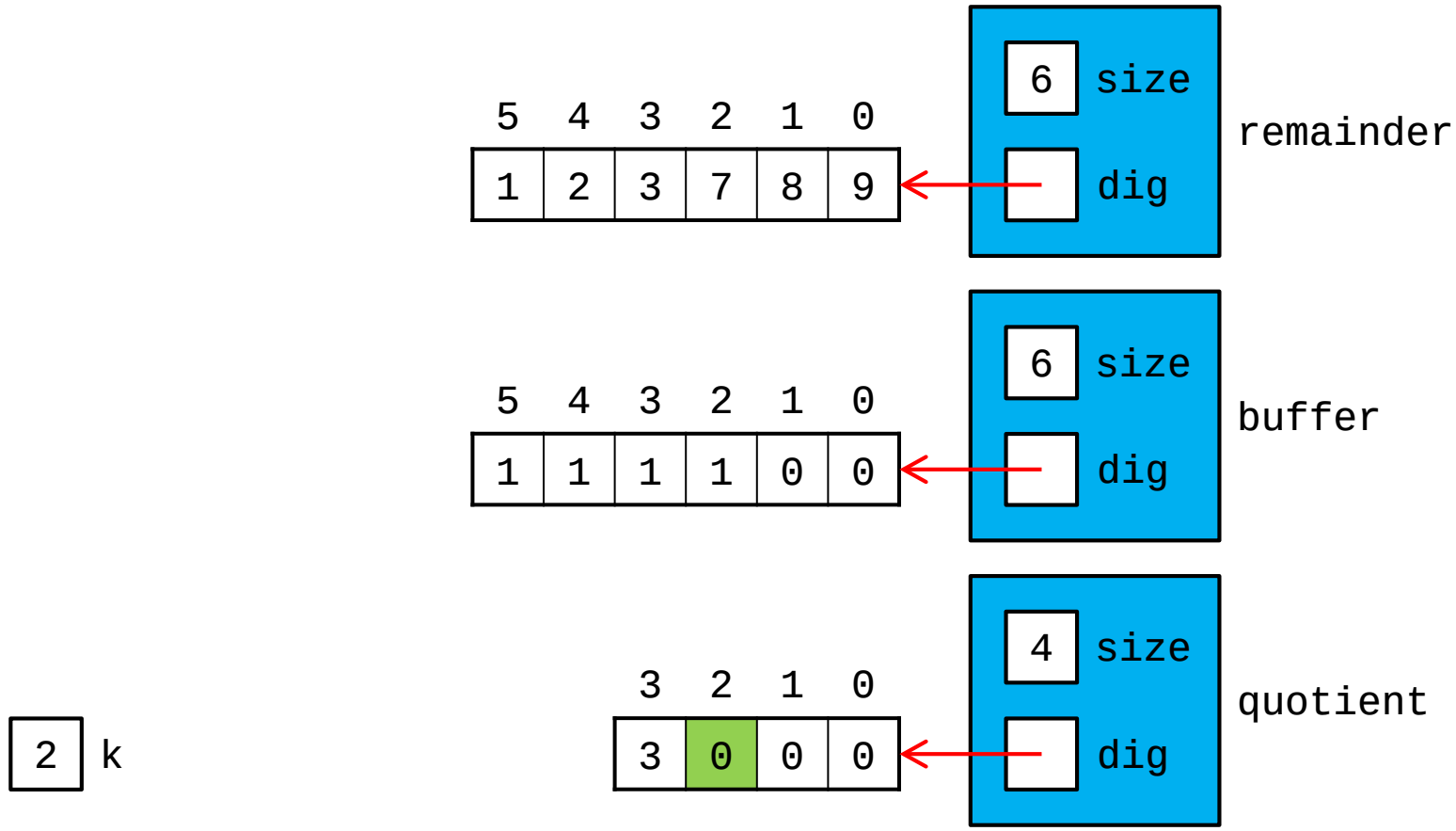
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



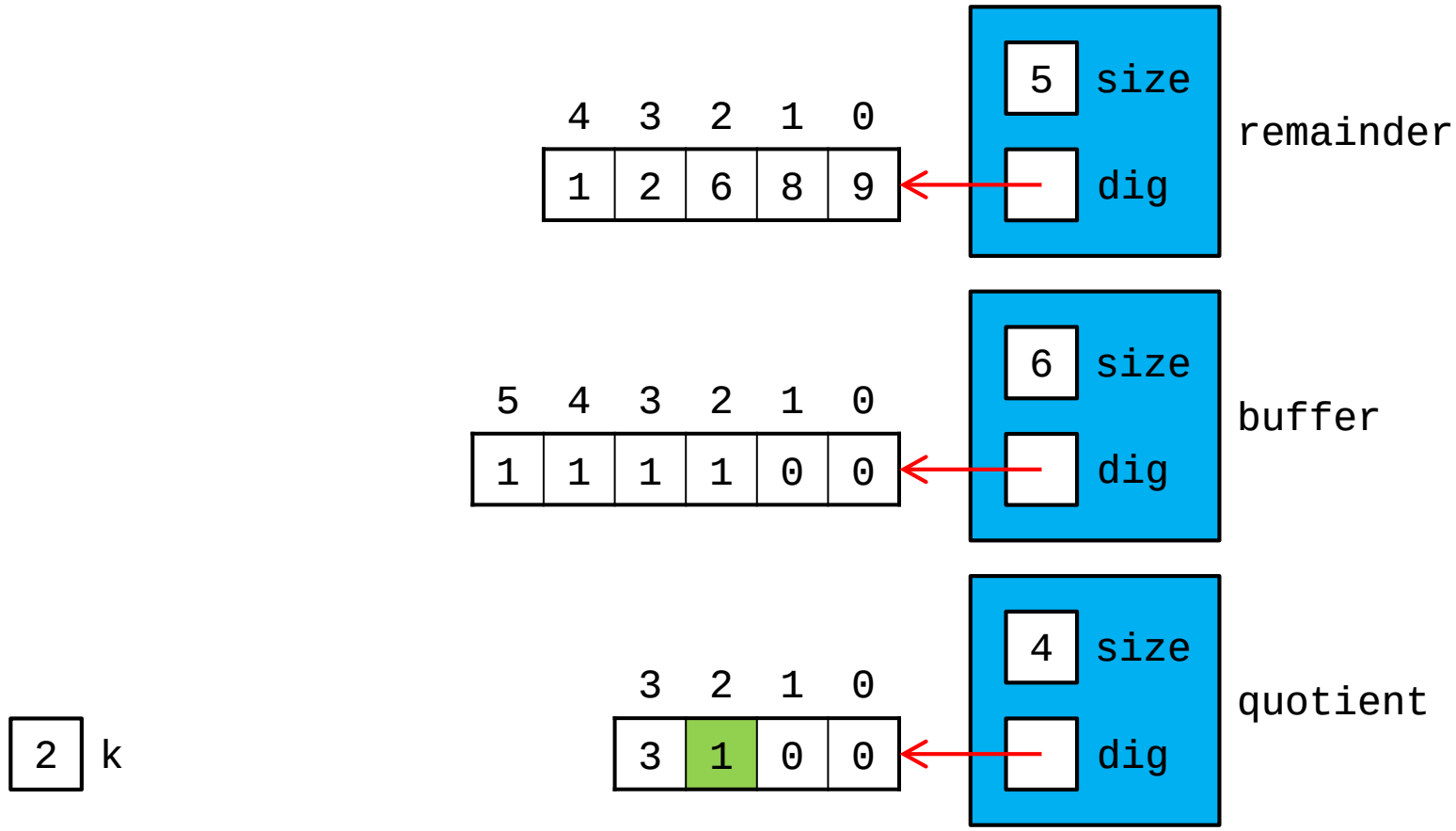
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



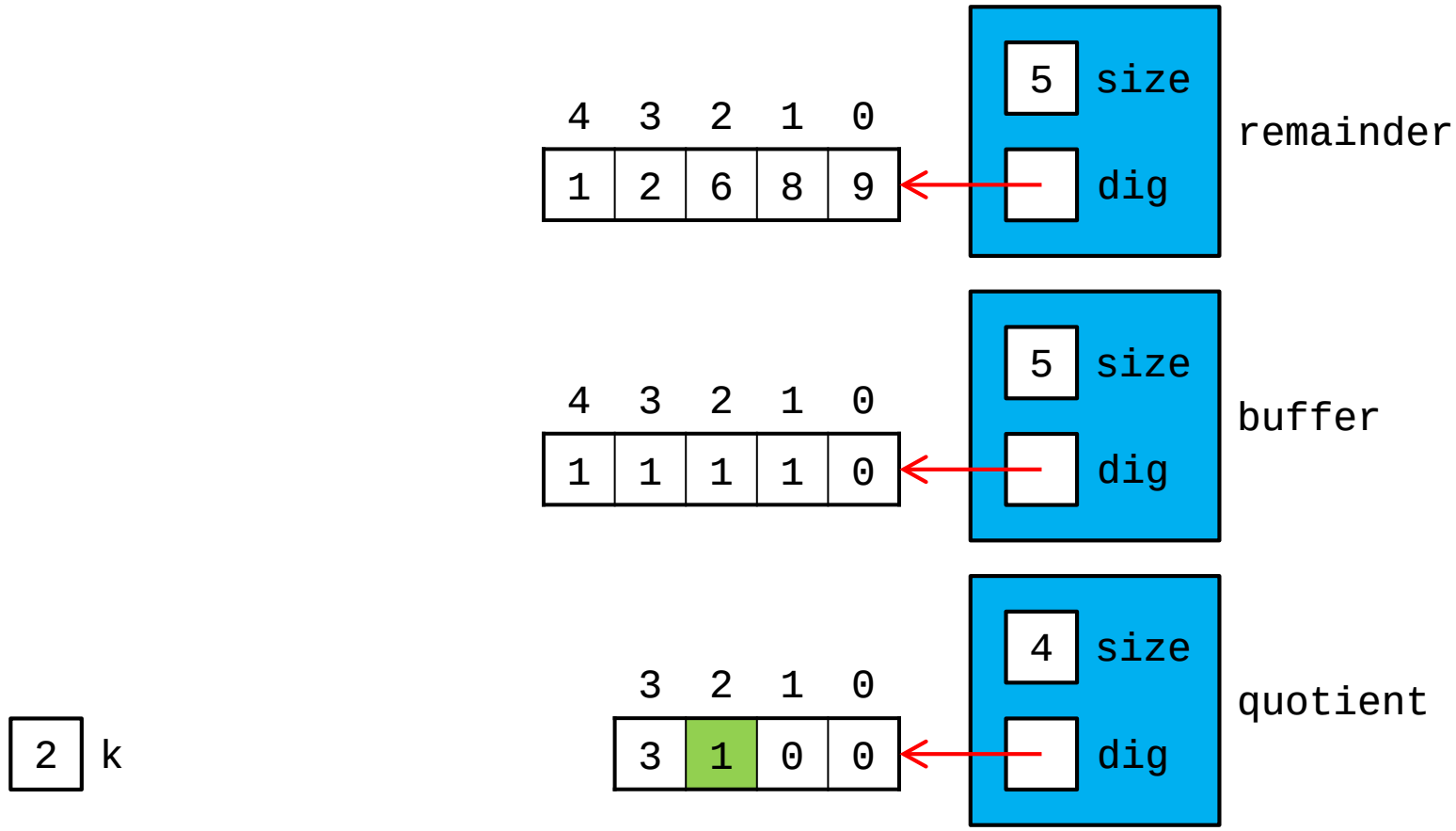
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



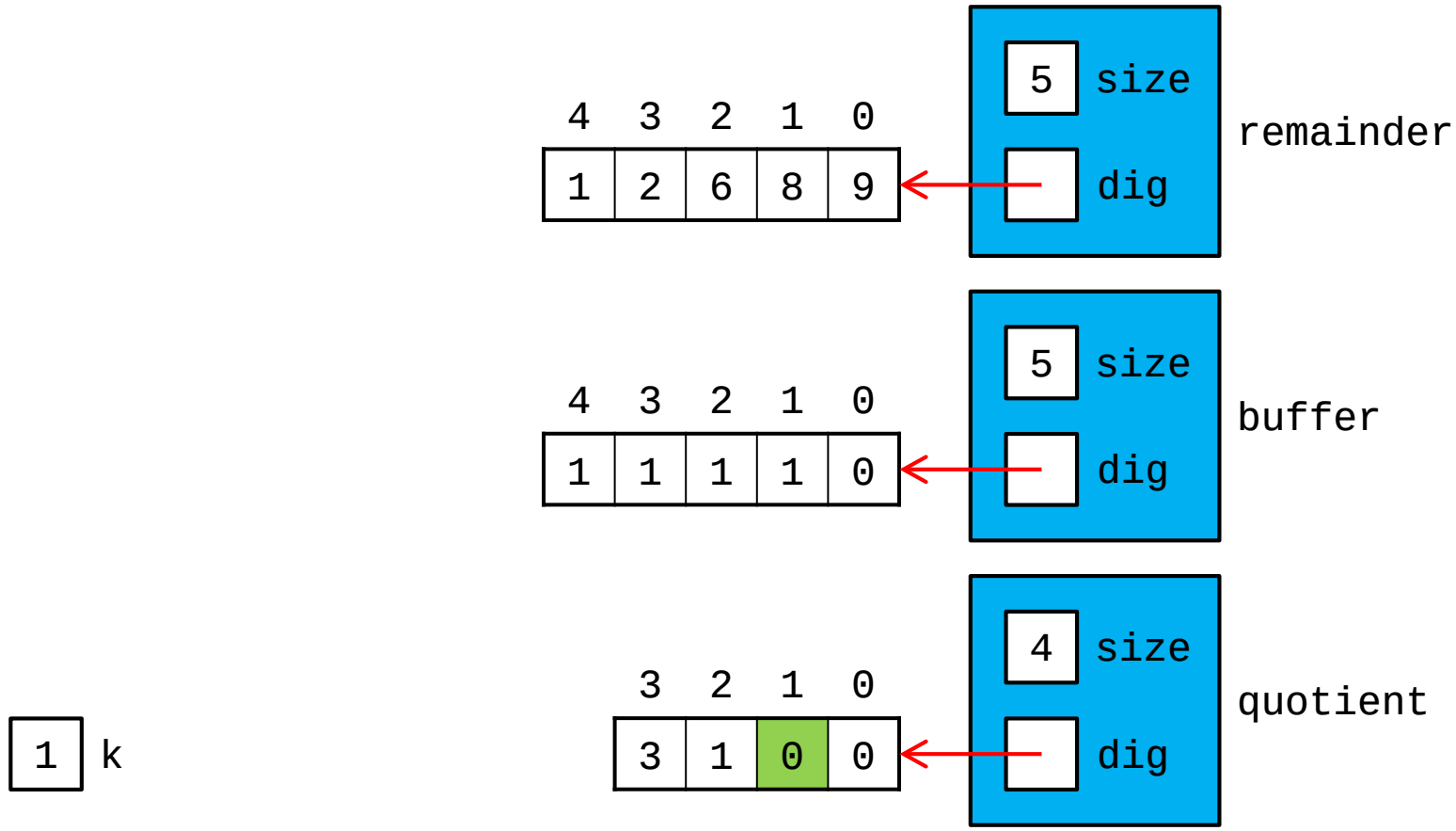
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



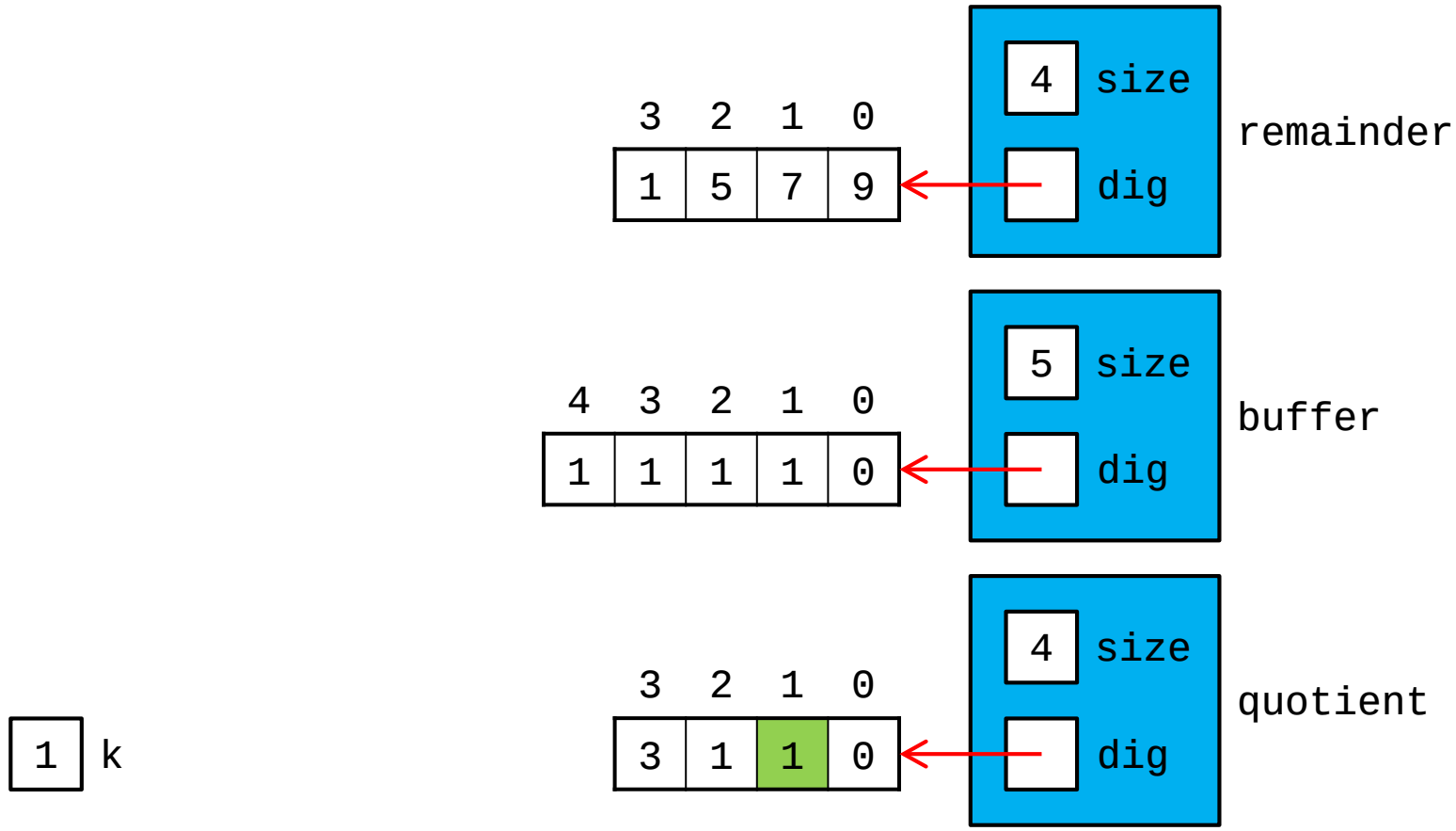
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



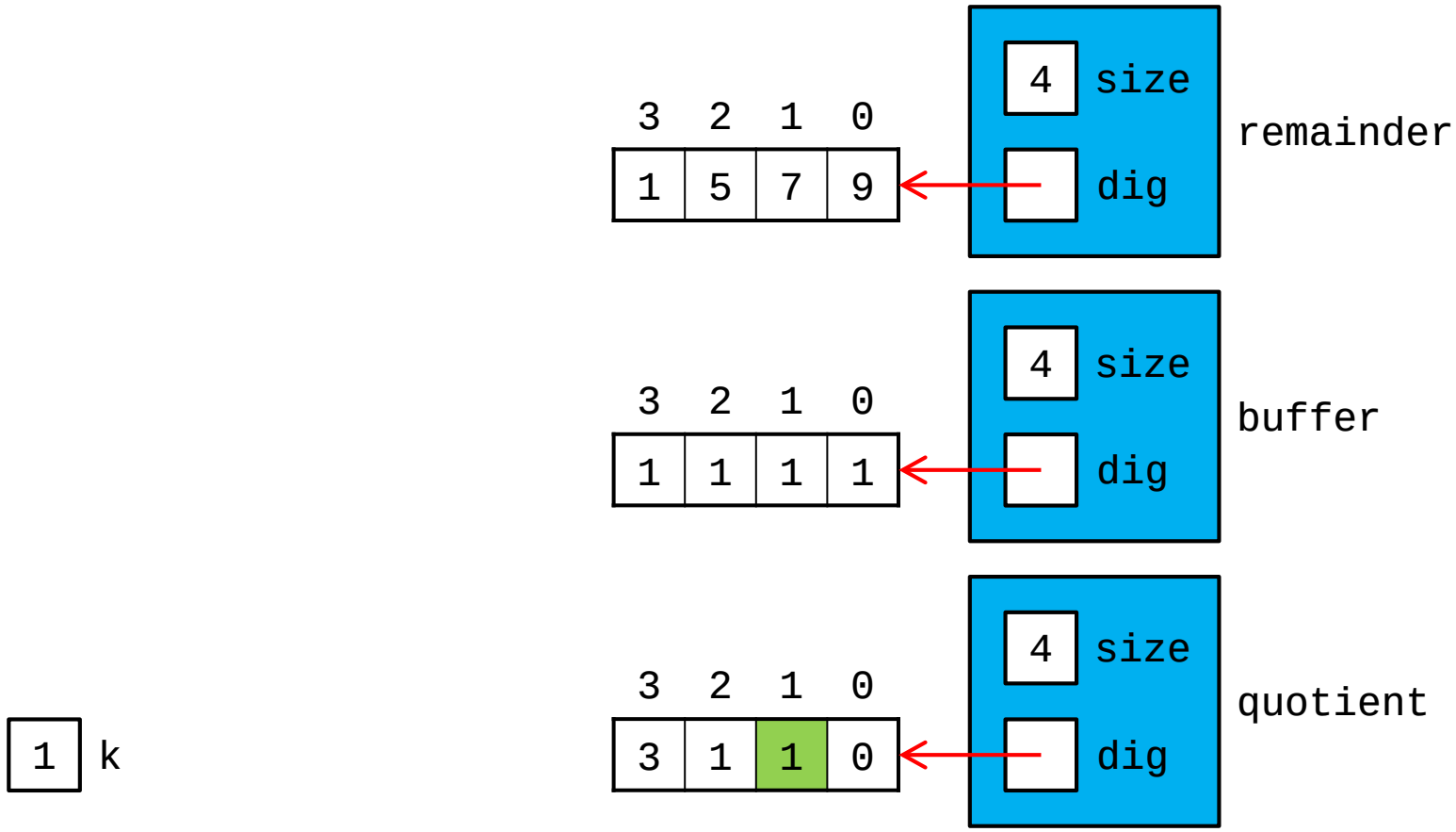
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



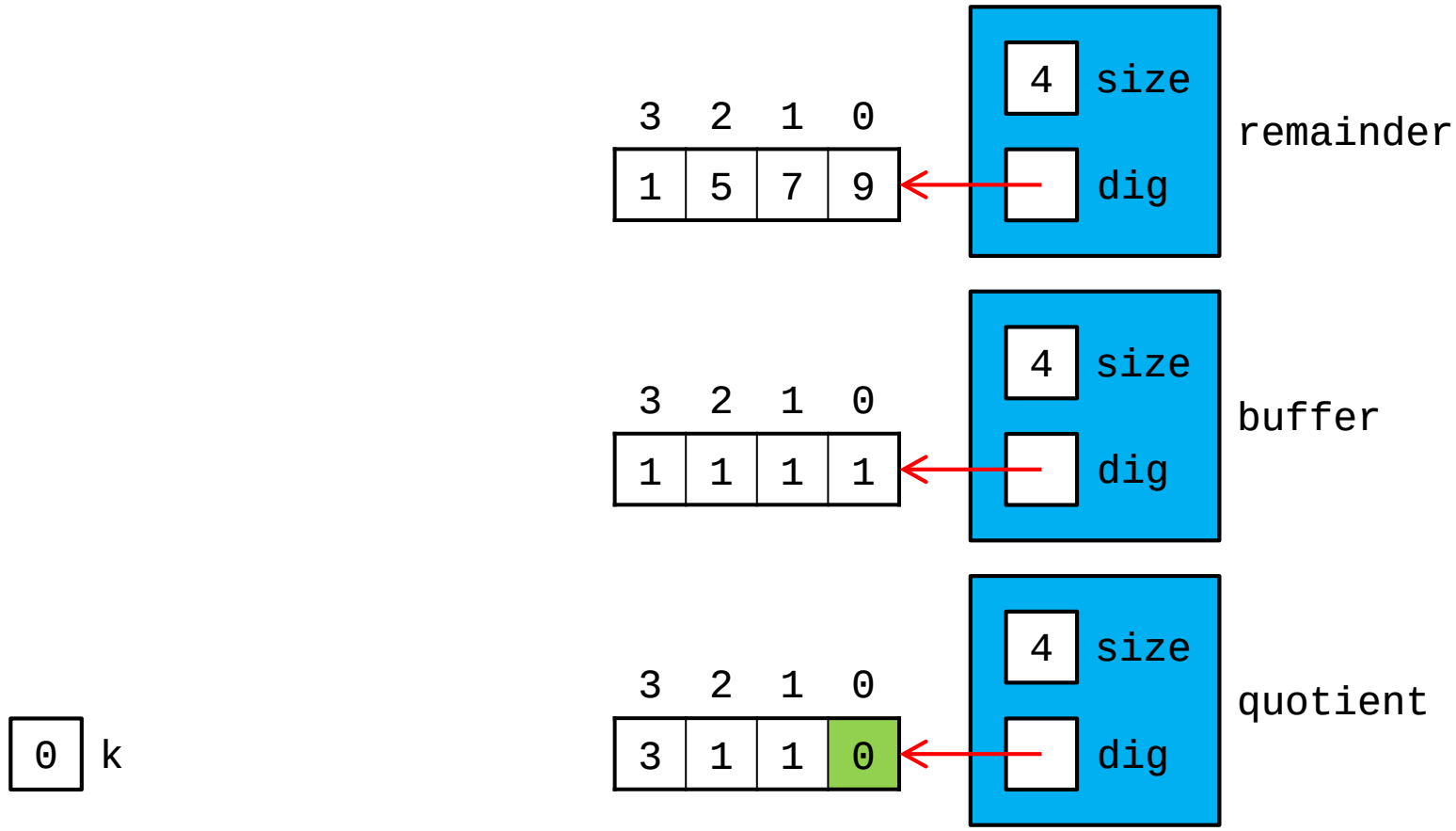
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



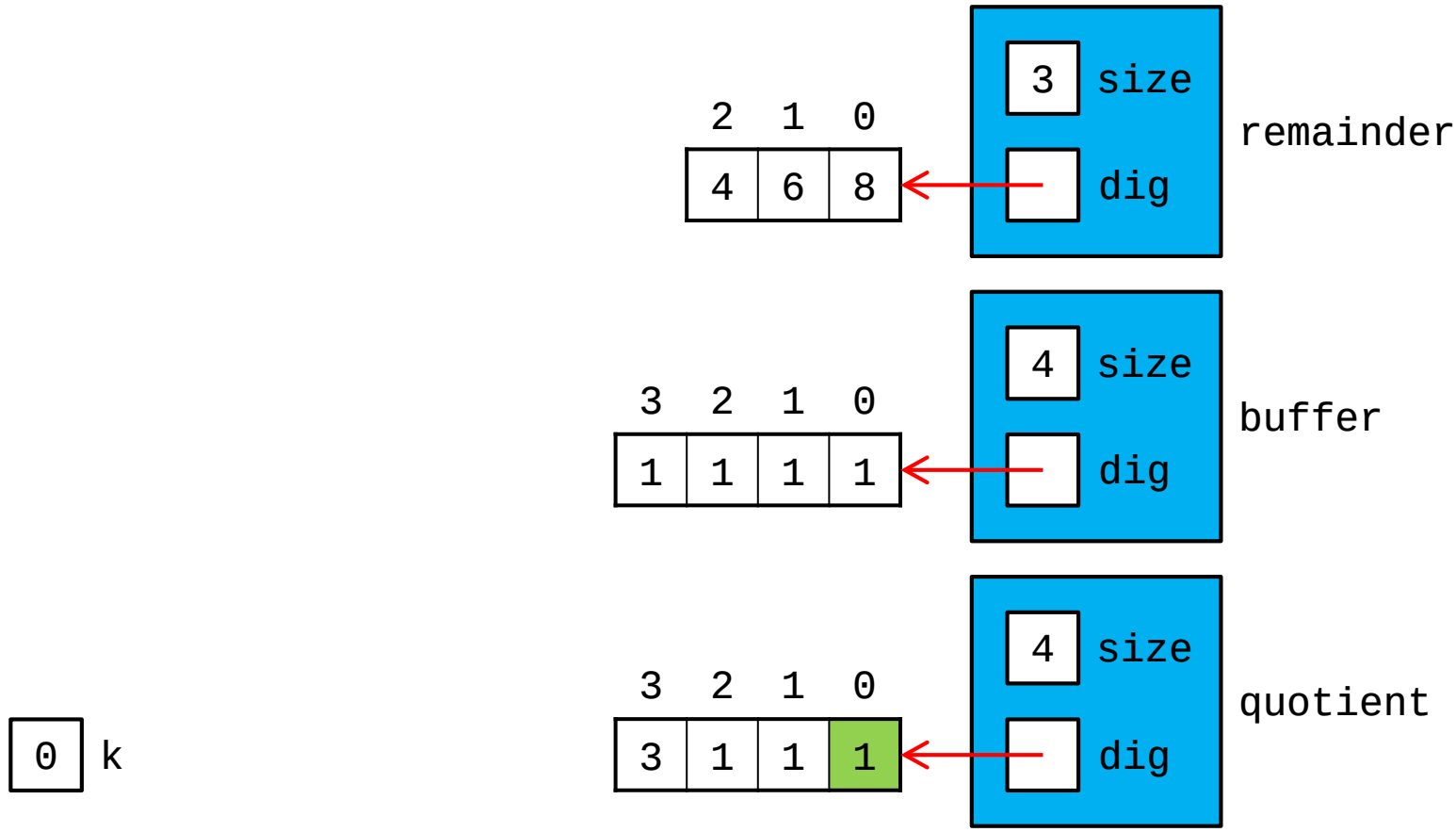
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



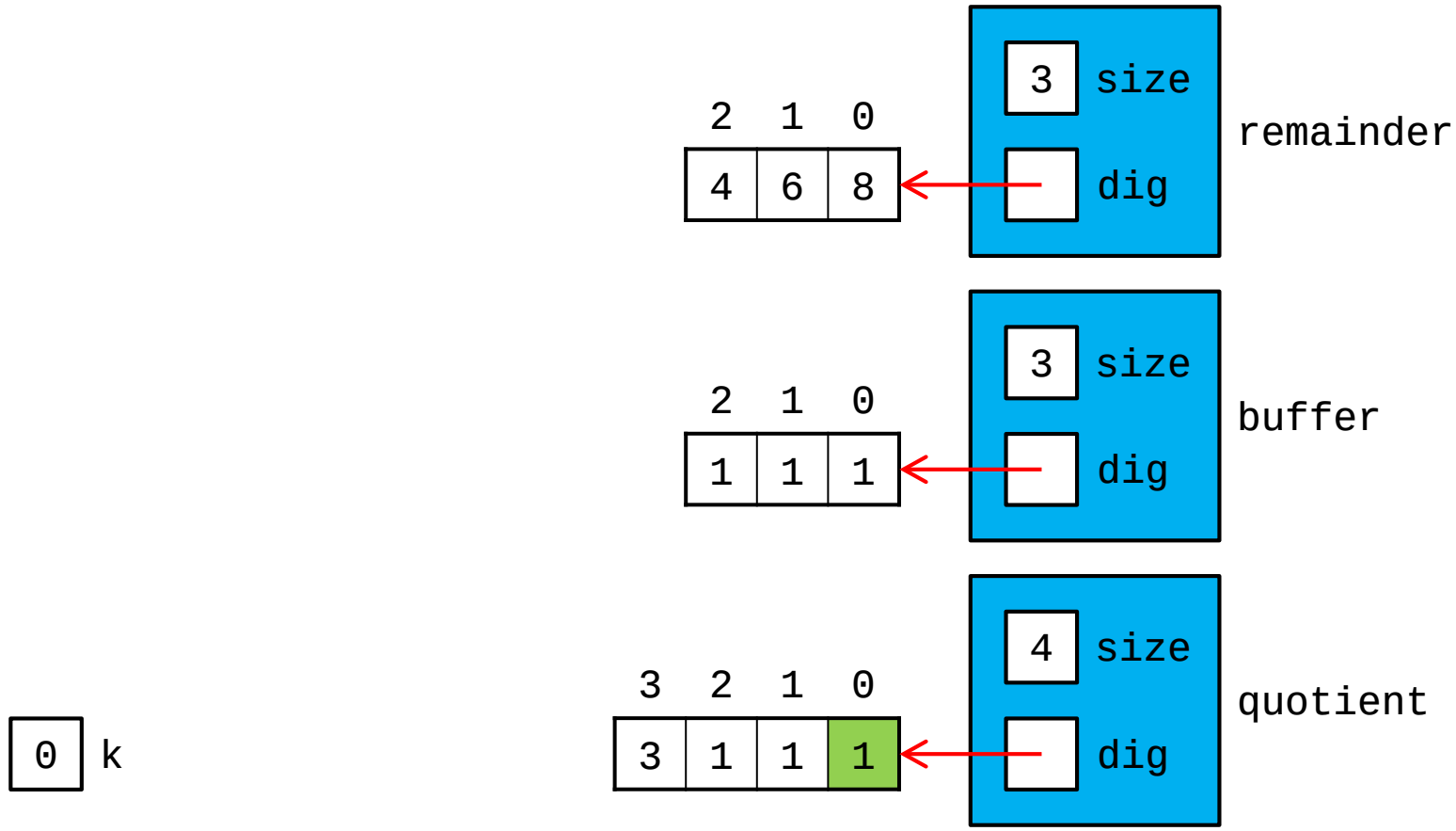
```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```



```
for( k = quotient.size - 1; k >= 0; k-- )
   while( remainder >= buffer )
      remainder -= buffer
      quotient[ k ]++
      if( remainder == 0 )
         return
   buffer = buffer / 10 // divideBy10( buffer )
```